

Making Sense of our coding Challenge - inspira

Documented solution to a coding challenge

By Donald Page



Table of Contents

1. Introduction	3
2. Goal and Objectives	3
2.1. Goal	3
2.2. Objectives	3
3. Research and Design	4
3.1. Research	4
3.1.1. For Loops	4
3.1.2. Switch Statements	5
4. Implementation	6
4.1. First Iteration	6
4.2. Second Iteration	7
4.3 Third Iteration	8
5. QA and Testing	10
6. Conclusion	11
7. Bibliography	12

1. Introduction

With the challenge set and accepted, this document aims to record and convey, the journey from start to finish. The following sections have been chosen, with the intention of clearly displaying all of the steps, which lead to the overall outcome of a working solution.

As with many coding solutions, a quick Google search provides a number of possibilities. With this in mind, to ensure full transparency, a bibliography has been provided. This includes links to all of the resources, which provided any inspiration and guidance for the proposed solution, outlined within this document.

2. Goal and Objectives

2.1. Goal

“ Write a program that prints the numbers from 1 to 100. But for multiples of three print the word “Sense” instead of the number and for the multiples of five print the word “Making”. For numbers which are multiples of both three and five print “Making Sense”. ”

2.2. Objectives

To achieve the goal (Sec 2.1), the following objectives have been set. The intention of these objectives, is to split the project down into logical steps. Once completed and documented, these steps aim to provide a working solution.

The objectives are set as follows:

1. Research and design a workable and complete solution.
2. Implement the design using a sensible language, within an iterative process.
3. Test the solution to the required standards, to prove the validity of the solution.
4. Document all progress in a professional and logical manor.

3. Research and Design

No rounded solution can be guaranteed, without a degree of research and this challenge was no different. This section therefore, outlays both the research conducted and the thoughts behind the design of the coded solution. All results of the research undertaken and referred to in this sections, can be found in the bibliography (Sec 7).

3.1. Research

Researching a solution to this challenge was relatively simple, but for the purpose of transparency, the results of this process have been documented below. The research conducted for this challenge was centered around which elements should be used to ensure an efficient outcome.

3.1.1. For Loops

Loops are used all the time, within code to ensure the same action is carried out, multiple times. However, given the parameter set within the goal of this challenge (Sec 2.1), of counting to 100. A for loop is a very good fit, as it allows for the action to be terminated after a set number of revolutions. Lending itself well, to carrying out a counting function, as shown below (Fig 1).

```
// example: sum numbers from 1 to 10
int total = 0;
for (int i=1; i<10; i++)
{
    total += i;
}
```

Fig 1. For loop counter

3.1.2. Switch Statements

This challenge required the program to carry out different actions, depending on the number produced by the counter (Sec 2.1). This can be achieved by a conditional statement, such as an “if statement”. However, because there are multiple actions which need implementing, a “switch statement” would be an efficient replacement to using multiple or nested if statements.

The switch statement is broken down into a condition, case and a default action. Enabling many different actions to be carried out, depending on the state of the provided condition. An example of a switch statement can be seen below (Fig 2).

```
switch ($i) {  
    case 0:  
        echo "i equals 0";  
        break;  
    case 1:  
        echo "i equals 1";  
        break;  
    case 2:  
        echo "i equals 2";  
        break;  
    default:  
        echo "i is not equal to 0, 1 or 2";  
}
```

Fig 2. Switch statement example

4. Implementation

Using the results of the research phase, the development process was ready to begin. As per the second objective (Sec 2.2), an iterative cycle was adopted. This is a very popular Agile methodology and is based around the idea of picking a starting point, for the first iteration. Then with each subsequent revolution, new elements are added until all the agreed parameters have been met.

Given the developers knowledge base, it was decided to complete the challenge using PHP, with an Apache server to display the results in the browser.

4.1. First Iteration

For the first iteration, it was decided to keep things simple, a basic proof of concept. Fig 3, shows the code used to to initiate a for loop counter, in this case with a limit set to 25. For every loop, the value held within \$x (variable) is echoed to screen.

```
<?php
// Starting value, Set limit value and action
for ($x = 1; $x <= 25; $x += 1) {
    echo $x . "...";
}
?>
```

Fig 3. For loop counter code snippet

As shown in Fig 4, this proves the basic loops works. With the loop running the code, everytime until the limit is hit.

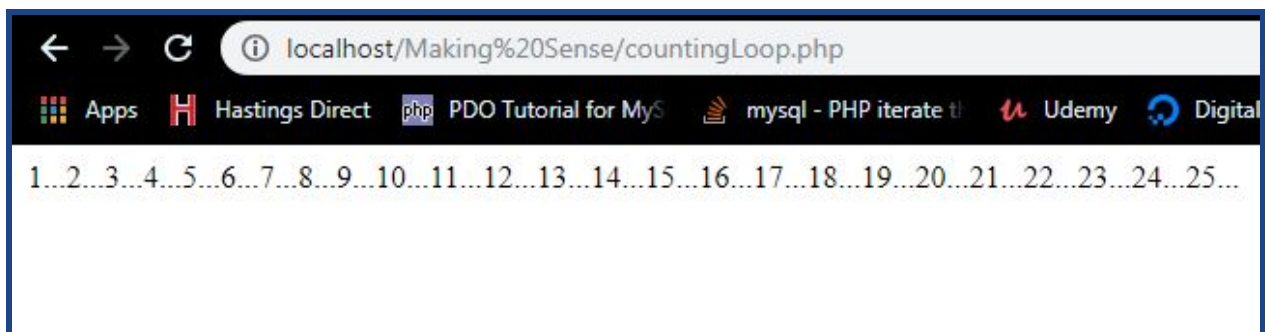


Fig 4. Screen print out for code in Fig. 3

4.2. Second Iteration

Aiming to build more functionality on top of the success of the first iteration, for the second it was decided to use a conditional statement to run within each loop. The condition asks whether the remainder of value held within $\$x$, after being divided by 3, is equal to 0. If this statement is true, then that value is replaced a string, held within $\$text$. This string value can be changed at any point, by simply replacing the string with a new value, as per the goal (Sec 2.1).

If the condition is false then the else statement will run, which simply echoes the value held within $\$x$, as before. The code for this, can be viewed in Fig 5.

```
<?php
// Simple For loop counter, set parameter of 25
for ($x = 1; $x <= 25; $x += 1) {
    // Using condition statement
    if ($x % 3 == 0) {           // If remainder of $x divided by 3 equals 0
        $text = "Making...";    // Set $text to contain a string
        echo $text;             // Echo $text to screen
    } else {                    // Otherwise..
        echo $x . "...";        // Echo $x to screen, with a separator
    }
}

?>
```

Fig 5. Counter loops with if statement

As before the screen output can be seen below (Fig 6.), with a string being echoed in the place of every number that is a multiple of three.



Fig 6. Screen print out for code in Fig.5

4.3 Third Iteration

Once again building functionality, the third iteration now features a switch statement (Fig 7). This is the proposed solution to enable multiple outcomes as per the original goal (Sec 2.1). As with the if statement, each case within the switch uses the modulus operator to compare the value of \$x. Depending on the argument given, the resulting output is exclusive to each case. Therefore, abiding by the parameters set in the original goal.

Using the default operation to set \$x to the value of \$text, when all case arguments are false, means that in normal conditions the value will be printed normally.

```
<?php
// Extending the limit of the counter to 100
for ($x = 1; $x <= 100; $x += 1) {
    // Replacing if with a switch for multiple outcomes
    switch ($x) {
        // Condition is the value of $x
        case $x % 15 == 0: // If remainder of $x divided by 15 equals 0
            // Run this code
            $text = "Making Sense" . "<br>"; // Set $text to contain
            // a string

            break;

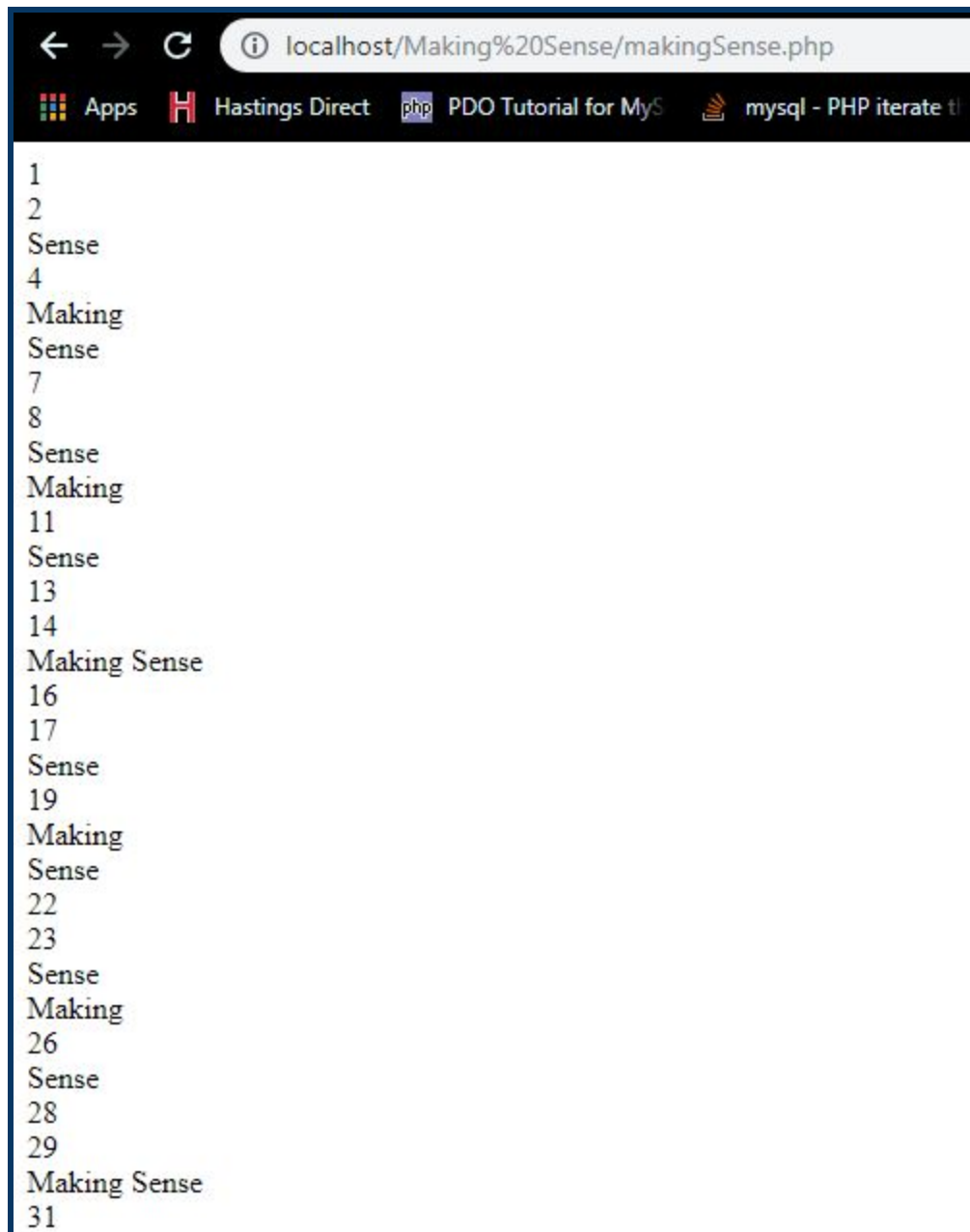
        case $x % 5 == 0: // If remainder of $x divided by 5 equals 0
            // Run this code
            $text = "Making" . "<br>"; // Set $text to contain string
            break;

        case $x % 3 == 0: // If remainder of $x divided by 3 equals 0
            // Run this code
            $text = "Sense" . "<br>"; // Set $text to contain a string
            break;

        default: // Default setting if no conditions are true
            $text = $x . "<br>"; // Set $text to the value of $x
    }
    echo $text; // each loop, echo $text to screen
}
?>
```

Fig 7. Code for third iteration and final solution

Fig 8 shows a clip from the print out to screen, when the code in Fig 7 is run in the browser. Clearly showing that when there is a multiple of 3, the word "Sense" is displayed. Equally, when there is a multiple of 5, the word "Making" is displayed. Finally, when there is a multiple of both 3 and 5, then "Making Sense" is displayed.



```
1
2
Sense
4
Making
Sense
7
8
Sense
Making
11
Sense
13
14
Making Sense
16
17
Sense
19
Making
Sense
22
23
Sense
Making
26
Sense
28
29
Making Sense
31
```

Fig 8. Snippet from the print out of the final solution

5. QA and Testing

Quality and Testing is a vital process within any development process. One of the advantages of adopting an iterative process, is that testing takes place at the end of each iterative cycle. Meaning that, before moving on to the next iteration, the code should perform the expected outcomes. This has been evidenced already, throughout Section 4.

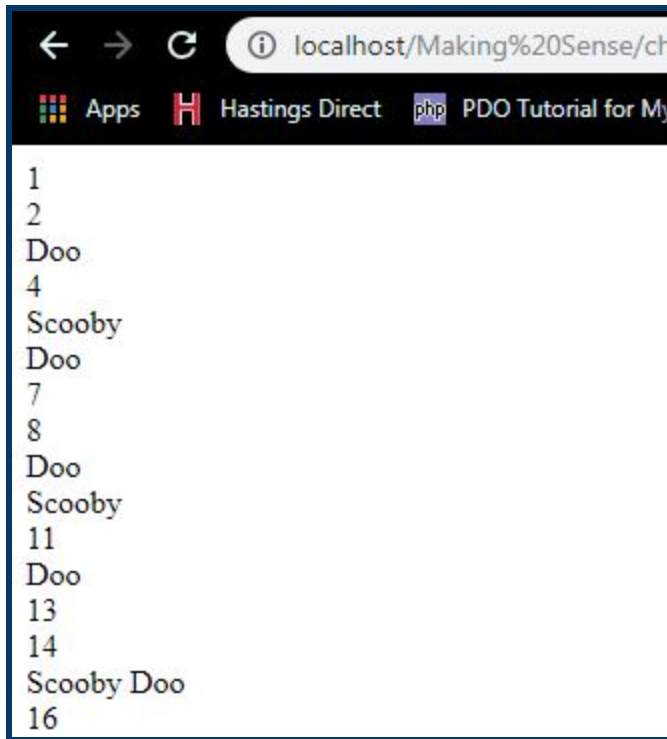
However, to demonstrate that storing the replacement text within a variable, allows the developer to easily manipulate what text is displayed. In this section, this process will be evidenced.

Figure 9 shows the the edited code for this test, where the string values have been changed.

```
<?php
// Extending the limit of the counter to 100
for ($x = 1; $x <= 100; $x += 1) {
    // Replacing if with a switch for multiple outcomes
    switch ($x) {
        // Condition is the value of $x
        case $x % 15 == 0: // If remainder of $x divided by 15 equals 0
            // Run this code
            $text = "Scooby Doo" . "<br>"; // Set $text to contain
            Break;                          // a string
        case $x % 5 == 0: // If remainder of $x divided by 5 equals 0
            // Run this code
            $text = "Scooby" . "<br>"; // Set $text to contain string
            break;
        case $x % 3 == 0: // If remainder of $x divided by 3 equals 0
            // Run this code
            $text = "Doo" . "<br>"; // Set $text to contain a string
            Break;
        default: // Default setting if no conditions are true
            $text = $x . "<br>"; // Set $text to the value of $x
    }
    echo $text; // each loop, echo $text to screen
}
?>
```

Fig 9. Code demonstrating the change to text values

The expected outcome of this test, is that the code will run as before. Except that, the value of the replacement string, will have changed as per Fig 9.



```
1
2
Doo
4
Scooby
Doo
7
8
Doo
Scooby
11
Doo
13
14
Scooby Doo
16
```

Fig 10. Screen print out after code changes

Fig 10 is the evidence that changing the values of the strings within each case, quickly and simply manipulates the outcome of the screen output.

6. Conclusion

When referring back to the the original goal (Sec 2.1), and comparing this against the evidence shown, for the final solution. It is clear that the proposed solution does deliver on these prescribed requirements.

On reflection however, if time was taken to complete a few more iterations, there could have been scope to add some functionality. For instance, using input boxes so that the user could easily manipulate the replacement strings. Therefore, excluding the need any code to be manually edited.

7. Bibliography

https://www.cs.utah.edu/~germain/PPS/Topics/for_loops.html

<http://php.net/manual/en/control-structures.switch.php>

https://www.w3schools.com/php/php_operators.asp

<http://www.unf.edu/~broggio/cis6516/classnotes/Chapter2.rr.ppt>

<https://gist.github.com/santyr/1155419>